Programming Languages and Programming Methodologies H02A3

http://people.cs.kuleuven.be/~gerda.janssens

online information on handouts, assignments and exercises

Prof. Gerda Janssens
Department of Computer Science
Celestijnenlaan 200A,
3001 Leuven(Heverlee)
tel 016/32 76 34
Gerda.janssens@cs.kuleuven.be

Who is here???

- Master of Artificial Intelligence
- Others

- Students : CW/ informatica K.U.Leuven
  course (Studie) Declarative Talen ???

Assignment week 1

Prepare chapter 2, namely sections 2.1 till 2.5

Question 1:

Indicate all the Prolog terms in the following lines of Prolog code:

```
move(state(P, onfloor, P, H), climb, state(P, onbox, 3.5, H)).
move(state(P, <--->, P, H),' onbox ',final).
haschild(barry) :- parent(barry, small('Caroline')).
```

Question 2:

2a) What is the role of the syntax in the context of programming languages?

2b) Can a Prolog sentence be ambiguous?

Assignment week 1

Question 3:

Did you have a previous course using GIL (Guided Independent Learning)? What were your experiences?

or Do you understand the concept of GIL? More questions?

Aim

Prolog is a well-known important representative of **logic** programming languages which form together with the functional languages the so-called **declarative** languages.

The course aims at introducing the Prolog language, at understanding substantial Prolog programs and at learning to write and debug Prolog programs.

Note that the theory and the use of logical formalisms is studied in the course Logic as a Foundation for AI (H02A2). Here we focus on the programming aspects and in that sense there is no dependence on LFAI.

Syllabus

Study of the syntax of Prolog and the procedural meaning of Prolog programs.

Study of data structures (lists, trees, ...), of data representation (as terms or as predicates), and their use in the context of some problem solving examples.

Study of builtins (a.o. for operators, arithmetic, input-output, findall,...), modules, cut, and negation.

Study of more advanced programming techniques such as open-ended structures, failure-driven iteration, meta interpreters, definite clause grammars.

Study of extensions of basic Prolog systems (possible candidates are constraint logic programming, programming with delays).

Teaching Activities

Guided independent learning (**GIL**): 13* 2 hours

every week the students get assignments to prepare some course material and to solve some related questions;
all this can be found on the web site of the course
every session: discussion (30 minutes) + additional exercises(1h30)

Exercises: 11* 2h30

exercises are solved by the students under the supervision of a teaching assistant; part of the sessions will be practical sessions on a Prolog system in one of the PC-rooms of the university, e.g. Celestijnenlaan 200C

Use of KULeuven Toledo system:

to enable e-discussions, but only if students are interested

Guided Independent Learning

- important didactic concept at the K.U.Leuven
- Students perform more researcher-like activities: students are expected to assume full responsibility for the learning process. Students are expected to
  - to acquire an in-depth understanding of the instructed goals
  - to select and engage in appropriate learning activities
  - call for additional help if the learning environment provides insufficient support.

Implications for students

- Students in general mention that they have to work very hard, even harder then before.
- But in order to learn a programming language you have to do it yourself and thus you have to work hard anyway.
- Moreover, you can use the moments of face-to-face contact to get answers to your problems and questions. It is important that you take this opportunity!!!

GIL in PLPM

We offer you the following learning environment:

- a stepwise approach to study the material and the opportunity to ask clarification and additional explanations and to make related exercises
  by the assignments and the questions
- the opportunity to make more exercises yourself on which you can get feedback from me or the teaching assistant
- the opportunity to use a Prolog system under supervision and to gain practical Prolog programming experience.

Important!!!

Exercises and **practical** sessions using a Prolog system are an indispensable part of the course.

You have the opportunity to get feedback, use it!!!

If you keep up with the material during the semester, you should be well prepared for the examination.

We will be using
SWI Prolog: http://www.swi-prolog.org/
and SICStus Prolog: http://www.sics.se/sicstus/

First Exercise Session

- Teaching assistants: Daan Fierens
- Wednesday 6/10/2010 14h00 200C.PC.A

  Celestijnenlaan 200C

Course Material

- Ivan Bratko, PROLOG programming for artificial Intelligence (third edition, 2001). Addison-Wesley Publishing Company.
- (The manual of a Prolog system).
- More books (for your information)
  - L. Sterling and E. Shapiro. The Art of Prolog: Advanced Programming Techniques, second edition, MIT Press, Cambridge, 1994.
  - R. O'Keefe. The Craft of Prolog, MIT Press, Cambridge, 1990.K.R. Apt. From Logic Programming to Prolog. International Series in Computer Science, Prentice Hall, 1997.

- R. Kowalski. Logic for problem solving. Artificial Intelligence Series, North Holland, 1974.
- P. Flach. Simply logical, Intelligent Reasoning by Example. John Wiley, 1994.
- P. Deransart, A. Ed-Dbali, and L. Cervoni. Prolog: The Standard. Springer-Verlag Berlin, 1996.
- K. Apt, V. Marek, M. Truszczynski, and D.S. Warren (eds). The Logic Programming Paradigm: A 25-Year Perspective. Springer, 1999.
- P. Van Hentenryck. Constraint Satisfaction in Logic Programming, Logic Programming Series, MIT Press, Cambridge, 1989.
- K.Marriott and P.J.Stukey. Programming with Constraints: An Introduction. MIT Press, 1998.

Interesting Links

- **Frequently Asked Questions (FAQ):** http://www.logic.at/prolog/faq/faq.pdf
- Links to online tutorials http://people.cs.kuleuven.be/~gerda.janssens/PLPM

More Practical Things

Prerequisites of this course:

- Some familiarity with algorithms and data structures.
- Knowledge of some programming language.

Course H02A2 Logic as a Foundation for Artificial Intelligence discusses logic formalisms and their theoretical background (syntax, semantics and proof procedures of logics).

Written exam focusing on exercises. Closed book. After the first semester.

Part I The Prolog Language

Chapter 1 An overview of Prolog 3
Chapter 2 Syntax and Meaning of Prolog Programs 25
Chapter 3 Lists, Operators, Arithmetic 61
Chapter 4 Using Structures: Example Programs 88
Chapter 5 Controlling Backtracking 114
Chapter 6 Input and Output 132
Chapter 7 More Built-in Procedures 147
Chapter 8 Programming Style and Technique 171
Chapter 9 Operations on Data Structures 197

Part II Prolog in Artificial Intelligence

Chapter 14 Constraint Logic Programming 319
Chapter 21 Language Processing with Grammar Rules 555
Chapter 23 Meta-programming 612

Questions???

1 An Overview of Prolog

overview basics mechanisms of Prolog through an example : clauses, backtracking and distinction between declarative and procedural meanings.

1.1 An example program: defining family relations

- symbolic, non-numeric computation
- objects (individuals H02A3) and relations
- Tom is a parent Bob objects? relation?

parent(tom,bob).

Family Database

```
parent(pam,bob).
parent(tom,bob).
parent(tom,liz).
parent(bob,ann).
parent(bob,pat).
parent(pat,jim).
```

6 clauses: actually facts
terminated by a full stop
instance of the parent relation
a relation is defined by the set of its instances

Prolog can be asked some questions
about the parent relation:

```
?- parent(bob,pat).      % fact!!
true
?- parent(liz,pat).      % no info
false.
?- parent(tom,ben).      % no info
false.
```

Family Database

```
parent(pam,bob).
parent(tom,bob).
parent(tom,liz).
parent(bob,ann).
parent(bob,pat).
parent(pat,jim).
```

Who is Liz's parent?

```
?- parent(X,liz).
X = tom ;
false.
```

Who are Bob's children?

```
?- parent(bob,X).
X = ann  ;
X= pat.
```

Who is the parent of whom?

```
?- parent(X,Y).
X=pam
Y=bob   ;
X= tom
Y= bob  <cr>
```

Family Database

```
parent(pam,bob).
parent(tom,bob).
parent(tom,liz).
parent(bob,ann).
parent(bob,pat).
parent(pat,jim).
```

Who is a grandparent of Jim?
no relation grandparent!!!
1. who is a parent of Jim? Assume Y.
2. who is a parent of Y? Assume X.

```
?- parent(Y,jim), parent(X,Y).
X =bob
Y = pat
```

Find a X and Y such that they satisfy the requirements:
parent(Y, jim) **and** parent(X, Y)
or equivalently:
parent(X, Y) **and** parent(Y, jim)

```
?- parent(X,Y), parent(Y, jim).
```

Who are Tom's grandchildren?

Family Database

```
parent(pam,bob).
parent(tom,bob).
parent(tom,liz).
parent(bob,ann).
parent(bob,pat).
parent(pat,jim).
```

Do Ann and Pat have a common parent?

1. who is a parent, X , of Ann?
2. Is (**this same**) X a parent of Pat?

```
?- parent(X,ann), parent(X,pat).
X = bob  ;
false.
```

Summary

- parent relation: pair in general n-tuples
- constants (also called atoms) : tom, ann
- variables: X, Y
- sequence of goals : conjunction of goals
- a goal is **satisfiable** if we get an answer (X = .../yes) the goal **succeeded**
- with answer no: the goal is **unsatisfiable** or the goal **failed**.
- if several answers satisfy the question then Prolog will find them all (as many of them as desired by the user).

1.2 Extending the program by rules

add information about the sex of the people.

- Unary relation expressing the property:

```
female(ann).
male (tom).
```

- relation associating the property male/female - person

```
sex(ann,feminine).
sex(tom,masculine).
```

1.2 Extending the program by rules

relation: offspring, namely inverse of parent/2

- by enumeration: offspring(bob,pam).
  for all parent/2 facts
  tedious, likely to introduce errors
- by defining the **rule**:
  for all X and Y, Y is an offspring of X
  if X is a parent of Y.

```
offspring(Y,X) :- parent(X,Y).
```

Prolog Rules

- A fact is something which is always true.
  parent(bob,ann).
- A rule specifies something that is true if some condition is satisfied:
  - the rhs of the rule: the condition part — the body
  - the lhs of the rule: the conclusion part — the head

  offspring(Y,X) :- parent(X,Y).

If the condition parent(X,Y) is true, then offspring(Y,X) is a logical consequence of it.

Querying Rules

?-offspring(liz,tom). (1)Apply the rule in terms of X and Y
offspring(Y,X) :- parent(X,Y).

apply it to liz and tom: Y has to be substituted by liz and X by tom
X and Y become instantiated: X = tom and Y = liz

also the rule becomes instantiated to:offspring(liz,tom) :- parent(tom,liz).

The condition part becomes: parent(tom,liz).
(2) Prolog tries to check whether the condition part is true.
(or in other words the original goal ?-offspring(liz,tom).
has been replaced by ?- parent(tom,liz).)
?-parent(tom,liz). (1) rule? a fact in the program.
Thus Prolog will answer: yes.

Summary

- Facts are clauses that have a head and an empty body.
- Questions only have a body.
- Rules have head and body.
- computation: a variable can be instantiated
- variables are universally quantified: "for all"
  hasachild(X) :- parent(X,Y).
  can be read as (alternative reading for Y (body only!!!)):
  - For all X and Y, if X is a parent of Y then X has a child
  - For all X, X has a child if there is some Y such that X is a parent of Y.

Important Observations

- Prolog = unification and backtracking
- Backtracking: Prolog uses backtracking to explore the whole search space and reports successes
- See : top-level interaction of Prolog systems
- Unification: no “assignment” to variables in Prolog;
  no X := X +1;
  but the system itself does bind the variables and uses unification (Bratko's matching).
- scope of the variables: limited to the query/the rule

1.3 A recursive rule definition

the **predecessor** relation:

1. immediate predecessor (in terms of parent)
2. indirect predecessor

1. for all X and Y,

X is a predecessor of Y if X is a parent of Y.

```
predecessor(X,Y) :- parent(X,Y).
```

2. indirect:

the parent of my parent (2 individuals)

the parent of (the parent of my parent) (3 individuals)

explicit as rules

Explicit Indirect Predecessor

```
predecessor(X,Y) :- parent(X,Y).
predecessor(X,Y) :-
    parent(X, Z),
    parent(Z,Y).
predecessor(X,Y) :-
    parent(X, Z1),
    parent(Z1,Z2),
    parent(Z2,Y).
predecessor(X,Y) :-
    parent(X, Z1),
    parent(Z1,Z2),
    parent(Z2,Z3),
    parent(Z3,Y).
```

program is lengthy
does not work in general
length of chain of people!!!

Indirect Predecessor: Recursively

For all X and Y, X is a predecessor of Y if
there is a Z such that
(1) X is a parent of Z and
(2) Z is a predecessor of Y.

```
predecessor(X,Y) :- parent(X,Y).
predecessor(X,Y):-
        parent(X,Z),
        predecessor(Z,Y).
```

recursive definition

essential in Prolog programming
(you can not live without them!!!!!)

Example

```
predecessor(X,Y) :- parent(X,Y).   % Rule pr1
predecessor(X,Y):-                 % Rule pr2
    parent(X,Z),
    predecessor(Z,Y).

?- predecessor(pam,X).
X = bob  ;
X = ann  ;
X = pat  ;
X = jim .
```

1.4 How Prolog answers questions

- A Prolog program is a set of procedures (relations), also called predicates.
- A procedure is a set of clauses about the same relation.
- a query/a goal ?- parent(X,ann), parent(ann,Y).
- satisfiable unsatisfiable
  yes no
  X = p1 Y = p2

Proof: Follows Logically

- Assume the relations/clauses are true.
- In order to satisfy a goal, Prolog has to demonstrate that the goal logically follows from the facts and the rules.
  mortal(X) :- man(X). %All men are mortal
  man(socrates).
  Thus, mortal(socrates).
- How does Prolog prove this? starting with a goal:
  ?- mortal(socrates).
  Look for a clause with "matching" head: mortal/1 and arguments that can be matched: X = socrates
  Check the condition ?- man(socrates).
  Has a matching fact.

Proofs

- Now suppose we want to know which persons are mortal:
  ?- mortal(X).
  Find matching clause, check condition ?- man(X).
  Has a matching fact: X = socrates.
- similarly for recursive rules: ?- predecessor(tom,pat).
  two matching rules have to be considered.
  1. ?- parent(tom,pat).
     this goal fails: no matching parent/2 fact

Proofs (cont.)

2. recursive clause for **?- predecessor(tom,pat).**
   instantiation: X = tom and Y = pat (not Z)
we obtain as goal the conditions of the recursive clause
?- parent(tom,Z), predecessor(Z,pat).

which of the two goals will Prolog take first?
left to right
thus, ?- parent(tom,Z) : can be matched with the fact
parent(tom, bob) and Z gets …
next, ?- predecessor(bob,pat).

What now??

?- predecessor(bob,pat).

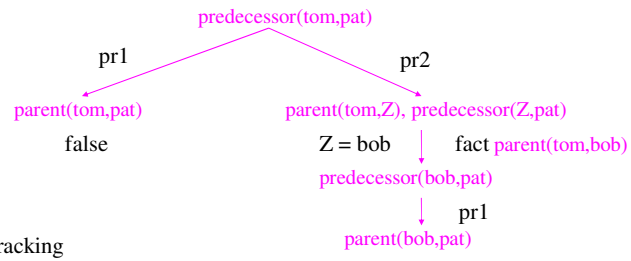
The two rules for predecessor/2 can be used again, but this use has nothing to do with previous application:
Prolog uses a new set of variables in the rule (renaming):

```
predecessor(X',Y') :- parent(X',Y').        % Rule pr1
predecessor(X',Y'):-                        % Rule pr2
        parent(X',Z'), predecessor(Z',Y').
```

Using pr1, ?- parent(bob,pat). with matching fact.
The goal is satisfied. (which goal(s)???)
The end??

Using pr2, ?- parent(bob,Z'), predecessor(Z',pat).
will fail. (try it!!)

Execution Trace



backtracking
which alternative clause ??
complete the unsuccessful branch

1.5 Declarative and procedural meaning of programs

- two levels of meaning (see also LFAI):
  declarative — WHAT
  procedural — HOW
- **declarative**: objects (individuals) and relations;
  clauses are expressing our knowledge
  reason about correctness and completeness of clauses.
  (semantics LFAI)
- **procedural**: how are the relations actually evaluated by the Prolog system (reasoning procedure LFAI)
- programmer should be concerned with declarative meaning; execution details should be left to Prolog.